WHAT IS CLAIMED IS:

1. A method for scheduling a plurality of tasks in a processing system having a plurality of defined resources, comprising:

identifying prerequisites for each task, the prerequisites representing all defined resources needed for that task to execute to completion;

in a task priority order, comparing the prerequisites for a particular task against a system state representing a current state of the defined resources until a task is identified for which the comparison reveals that the prerequisites for the identified task are currently available; and

dispatching the identified task.

2. The method as recited in claim 1, further comprising:

identifying one or more priority task paths in a task processing sequence; and

arranging the task priority order such that the tasks in the priority task paths have a higher priority than other tasks.

3. The method as recited in claim 1, further comprising:

identifying one or more fast paths in a task processing sequence; and

for a given fast path, arranging the task priority order such that the tasks in the given fast path have a higher priority than other tasks.

4. The method as recited in claim 3, further comprising:

identifying a processing order of the tasks in the given fast path; and

arranging the task priority order such that a last task in the given fast path has a higher priority than a first task in the given fast path.

5. The method as recited in claim 3, further comprising:

identifying a processing order of the tasks in the given fast path; and

arranging the task priority order such that a first task in the given fast path has a higher priority than a last task in the given fast path.

6. The method as recited in claim 1, further comprising representing each defined resource by one or more resource flags that provide information about that resource.

7. The method as recited in claim 6, further comprising storing the prerequisites for each task as a collection of resource flags known as a prerequisite row and storing the system state as a collection of resource flags.

8. The method as recited in claim 7, further comprising comparing the prerequisite row for the particular task against the system state by performing a Boolean AND of the prerequisite row and the system state.

9. The method as recited in claim 7, further comprising:

creating a prerequisite table, the prerequisite table having one prerequisite row for each instantiated task arranged in descending order according to the task priority order, and having one column for each resource flag; and

in a descending row order, performing a comparison of the prerequisite row for the task in a given row against the system state.

10. The method as recited in claim 9, further comprising:

creating a decision tree from the prerequisite table according to the task priority order, the decision tree having one leaf for each instantiated task, one node for each resource, and one edge for each resource flag; and

traversing the decision tree in accordance with the availability of resources.

11. The method as recited in claim 1, further comprising updating the system state after resources are actually used.

12. The method as recited in claim 11, wherein after the system state has been updated, the method further comprises implementing fair-share scheduling by resuming the comparison of prerequisite rows against the updated system state at a next row in the prerequisite table.

13. The method as recited in claim 11, wherein once the system state has been updated, the method further comprises implementing priority scheduling by resuming the comparison of prerequisite rows against the updated system state at a first row in the prerequisite table.

14. The method as recited in claim 11, further comprising:

grouping consecutive prerequisite rows in the prerequisite table into one or more blocks;

when a task is identified and scheduled for dispatch, identifying the block to which the identified task belongs; and

once the identified task is dispatched and the system state is updated, resuming the comparison at a next row within the identified block, or resuming the comparison at a first row of the prerequisite table if the dispatched task was a last task in the identified block.

15. The method as recited in claim 1, further comprising:

creating one or more priority flags representing various priority levels as defined resources in the processing system; and

adding priority flag resources to the prerequisites of selected tasks in accordance with priority levels desired for those tasks;

wherein the priority flag resources effectively change the task priority order as the prerequisites for the selected tasks are compared against the system state.

16. The method as recited in claim 1, the defined resources including one or more queues, the method further comprising:

changing a depth of one or more queues to alter the scheduling of tasks having those queues as prerequisites.

17. The method as recited in claim 1, the defined resources including one or more buffer pools, the method further comprising:

changing a depth of one or more buffer pools to alter the scheduling of tasks having those buffer pools as prerequisites.

18. The method as recited in claim 9, further comprising:

storing the prerequisite rows in M N-bit words, each bit in each N-bit word representing a resource flag and each resource flag being either asserted to indicate that the resource flag is a prerequisite of the task represented by the prerequisite row or deasserted to indicate that the resource flag is not a prerequisite of that task;

storing the system state in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted or deasserted in accordance with the current state of the processing system;

storing an indicator of the last N-bit word containing an asserted resource flag;

comparing the prerequisites and the system state of successive N-bit words only up to the last N-bit word containing an asserted resource flag; and

determining whether the prerequisites for the identified task are currently available using only the compared N-bit words.

19. The method as recited in claim 18, further comprising:

arranging the resource flags that make up the prerequisite rows and the system state so that when the prerequisite rows are stored in the M-bit words, the number of asserted resource flags appearing in the first M-bit word is maximized.

20. The method as recited in claim 9, further comprising:

storing the prerequisite rows in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted to indicate that the resource flag is a prerequisite of the task represented by the prerequisite row or deasserted to indicate that the resource flag is not a prerequisite of that task;

storing the system state in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted or deasserted in accordance with the current state of the processing system;

comparing the prerequisites and the system state of successive N-bit words only as long as the comparison indicates that the prerequisites for the identified task are currently available.

21. The method as recited in claim 2, further comprising:

modifying a scope of the prerequisites for the tasks in a particular priority task path to adjust a processing speed of that priority task path.

22. In a processing system having a plurality of defined resources, the processing system for executing a plurality of tasks, a computer program for scheduling the plurality of tasks, the computer program being stored on a machine readable medium and executable to perform acts comprising:

identifying prerequisites for each task, the prerequisites representing all defined resources needed for that task to execute to completion;

in a task priority order, comparing the prerequisites for a particular task against a system state representing a current state of the defined resources until a task is identified for which the comparison reveals that the prerequisites for the identified task are currently available; and

scheduling the identified task for dispatch.

23. The computer program as recited in claim 22, further executable to perform acts comprising

identifying one or more priority task paths in a task processing sequence; and

arranging the task priority order such that the tasks in the priority task paths have a higher priority than other tasks.

24. The computer program as recited in claim 22, further executable to perform acts comprising:

identifying one or more fast paths in a task processing sequence; and

for a given fast path, arranging the task priority order such that the tasks in the given fast path have a higher priority than other tasks.

25. The computer program as recited in claim 24, further executable to perform acts comprising:

identifying a processing order of the tasks in the given fast path; and

arranging the task priority order such that a last task in the given fast path has a higher priority than a first task in the given fast path.

26. The computer program as recited in claim 24, further executable to perform acts comprising:

identifying a processing order of the tasks in the given fast path; and

arranging the task priority order such that a first task in the given fast path has a higher priority than a last task in the given fast path.

27. The computer program as recited in claim 22, further executable to perform acts comprising representing each defined resource by one or more resource flags that provide information about that resource.

28. The computer program as recited in claim 27, further executable to perform acts comprising storing the prerequisites for each task as a collection of resource flags known as a prerequisite row and storing the system state as a collection of resource flags.

29. The computer program as recited in claim 28, further executable to perform acts comprising comparing the prerequisite row for the particular task against the system state by performing a Boolean AND of the prerequisite row and the system state.

30. The computer program as recited in claim 28, further executable to perform acts comprising:

creating a prerequisite table, the prerequisite table having one prerequisite row for each instantiated task arranged in descending order according to the task priority order, and having one column for each resource flag; and

in a descending row order, performing a comparison of the prerequisite row for the task in a given row against the system state.

31. The computer program as recited in claim 30, further executable to perform acts comprising:

creating a decision tree from the prerequisite table according to the task priority order, the decision tree having one leaf for each instantiated task, one node for each resource, and one edge for each resource flag; and

traversing the decision tree in accordance with the availability of resources.

32. The computer program as recited in claim 22, further executable to perform acts comprising updating the system state as resources are actually used.

33. The computer program as recited in claim 32, wherein after the system state has been updated, the computer program is further executable to perform acts comprising fair-share scheduling by resuming the comparison of prerequisite rows against the updated system state at a next row in the prerequisite table.

34. The computer program as recited in claim 32, wherein once the system state has been updated, the computer program is further executable to perform acts comprising priority scheduling by resuming the comparison of prerequisite rows against the updated system state at a first row in the prerequisite table.

35. The computer program as recited in claim 32, further executable to perform acts comprising:

grouping consecutive prerequisite rows in the prerequisite table into one or more blocks;

when a task is identified and scheduled for dispatch, identifying the block to which the identified task belongs; and

once the identified task is dispatched and the system state is updated, resuming the comparison at a next row within the identified block, or resuming the comparison at a first row of the prerequisite table if the dispatched task was a last task in the identified block.

36. The computer program as recited in claim 22, further executable to perform acts comprising:

creating one or more priority flags representing various priority levels as defined resources in the processing system; and

adding priority flag resources to the prerequisites of selected tasks in accordance with priority levels desired for those tasks;

wherein the priority flag resources effectively change the task priority order as the prerequisites for the selected tasks are compared against the system state.

37. The computer program as recited in claim 22, the defined resources including one or more queues, the computer program further executable to perform acts comprising:

changing a depth of one or more queues to alter the scheduling of tasks having those queues as prerequisites.

38. The computer program as recited in claim 22, the defined resources including one or more buffer pools, the computer program further executable to perform acts comprising:

changing a depth of one or more buffer pools to alter the scheduling of tasks having those buffer pools as prerequisites.

39. The computer program as recited in claim 30, further executable to perform acts comprising:

storing the prerequisite rows in M N-bit words, each bit in each N-bit word representing a resource flag and each resource flag being either asserted to indicate that the resource flag is a prerequisite of the task represented by the prerequisite row or deasserted to indicate that the resource flag is not a prerequisite of that task;

storing the system state in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted or deasserted in accordance with the current state of the processing system;

storing an indicator of the last N-bit word containing an asserted resource flag;

comparing the prerequisites and the system state of successive N-bit words only up to the last N-bit word containing an asserted resource flag; and

determining whether the prerequisites for the identified task are currently available using only the compared N-bit words.

40. The computer program as recited in claim 39, further executable to perform acts comprising:

arranging the resource flags that make up the prerequisite rows and the system state so that when the prerequisite rows are stored in the M-bit words, the number of asserted resource flags appearing in the first M-bit word is maximized.

41. The computer program as recited in claim 30, further executable to perform acts comprising:

storing the prerequisite rows in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted to indicate that the resource flag is a prerequisite of the task represented by the prerequisite row or deasserted to indicate that the resource flag is not a prerequisite of that task;

storing the system state in M N-bit words, each bit in each N-bit word representing a resource flag, each resource flag being either asserted or deasserted in accordance with the current state of the processing system;

comparing the prerequisites and the system state of successive N-bit words only as long as the comparison indicates that the prerequisites for the identified task are currently available.

42. The computer program as recited in claim 23, further executable to perform acts comprising:

modifying a scope of the prerequisites for the tasks in a particular priority task path to adjust a processing speed of that priority task path.

43. A host bus adapter (HBA) comprising the machine readable medium of claim 22.

44. The HBA of claim 43, further comprising an Internet Small Computer System Interface (iSCSI) or fibre channel (FC) controller circuit.

45. A host computer comprising the HBA of claim 44.

46. A storage area network (SAN) comprising the host computer of claim 45, wherein an iSCSI or a FC network is coupled to the iSCSI or FC controller circuit, respectively, and one or more storage devices are coupled to the iSCSI or FC network.